EcomLabs

STEP-BY-STEP GUIDE

A custom field in Shopify checkout

A complete checkout UI extension with a purchase-reference field: the config file, the component, the cart metafield that is copied to the order and the steps to deploy it. With the Shopify Plus rule and a checklist.

FOR	Shopify apps that add a field to the checkout of a Plus store
TESTED WITH	Code validated with Shopify's tools (@shopify/ui-extensions 2026.1.1 and the Admin GraphQL schema). Not run in a store
UPDATED	November 12, 2026
DOCS	shopify.dev/docs/apps/build/checkout/fields-banners/add-field
ARTICLE	ecomlabs.dev/g/campo-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will build
Step 01	Create the extension
Step 02	shopify.extension.toml
Step 03	The component
Step 04	The order definition
Step 05	Deploy and activate
When you finish	Checklist

BEFORE YOU START

What you will build

A checkout UI extension that renders a text field, "purchase reference", right after the contact block in checkout, and saves what the shopper types in a cart metafield. With an order metafield definition, that value is copied to the order. You need a **Shopify Plus** store (extensions on information, shipping and payment only exist there) and **Shopify CLI 3.85.1 or higher**, the version the official tutorial asks for.

STEP 01

Create the extension

Inside your app folder:

```
shopify app generate extension --template checkout_ui --name po-reference
```

The CLI creates `shopify.extension.toml`, `src/Checkout.jsx`, `locales/` and `package.json`. To see the extension while you develop, run `shopify app dev` and press `p`. If you edit the `.toml`, restart the command.

STEP 02

shopify.extension.toml

The CLI generates the `uid` and the API version; keep whatever it writes. This is the content that matters:

```
api_version = "2026-10"

[[extensions]]
name = "po-reference"
handle = "po-reference"
type = "ui_extension"

[[extensions.targeting]]
module = "./src/Checkout.jsx"
target = "purchase.checkout.contact.render-after"

[[extensions.targeting.metafields]]
namespace = "my_app"
key = "po_reference"
```

The `target` sets the place: for a shipping field use `purchase.checkout.shipping-option-list.render-after`. `namespace` and `key` are yours, but they must be repeated exactly in the code and in the order definition (Step 04).

STEP 03

The component

Replace `src/Checkout.jsx`. We validated it with Shopify's validator for checkout extensions, on the `purchase.checkout.contact.render-after` target:

```
import '@shopify/ui-extensions/preact';
import {render} from 'preact';
import {useState} from 'preact/hooks';

const NAMESPACE = 'my_app';
const KEY = 'po_reference';

export default function extension() {
  render(<Extension />, document.body);
}

function Extension() {
  const {appMetafields, applyMetafieldChange, instructions, i18n} = shopify;

  // The value saved earlier in this checkout, so the field survives a reload.
  const saved = appMetafields.value.find(
    (entry) =>
      entry.target.type === 'cart' &&
      entry.metafield.namespace === NAMESPACE &&
      entry.metafield.key === KEY,
  );
  const [value, setValue] = useState(String(saved?.metafield.value ?? ''));
  const [failed, setFailed] = useState(false);

  // Accelerated checkout (Apple Pay, Google Pay) can turn this off.
  if (!instructions.value.metafields.canSetCartMetafields) {
    return null;
  }

  async function save() {
    const text = value.trim();
    const result = await applyMetafieldChange(
      text === ''
        ? {type: 'removeCartMetafield', namespace: NAMESPACE, key: KEY}
        : {
            type: 'updateCartMetafield',
            metafield: {
              namespace: NAMESPACE,
              key: KEY,
              type: 'single_line_text_field',
              value: text,
            },
          },
    );
```

```
    setFailed(result.type === 'error');
  }

  function updateValue(event) {
    setValue(event.currentTarget.value);
  }

  return (
    <s-text-field
      label={i18n.translate('poLabel')}
      value={value}
      error={failed ? i18n.translate('poError') : undefined}
      onInput={updateValue}
      onBlur={save}
    />
  );
}
```

The `poLabel` and `poError` strings go in `locales/en.default.json`, for example `{"poLabel": "Purchase order reference (optional)", "poError": "We couldn't save this. Try again."}`, plus one file per language, such as `locales/es.json`. The field saves on blur (`onBlur`), not on every keystroke: too many changes can rate limit the extension. Do not use `applyAttributeChange`, which the API reference marks deprecated. With `shipping-option-list.render-after`, which renders once per purchase type, return `null` unless `shopify.target.value.groupType` is `'oneTimePurchase'`, as the official example does.

STEP 04

The order definition

Shopify copies the cart metafield to the order when the order is created, only if an order metafield definition exists with the same namespace and key and the *cart to order copyable* capability. Two ways to create it.

In the admin: create an **order** metafield definition, single line text, namespace `my_app`, key `po_reference`, and turn on *Copy values from cart metafields*.

With the API (mutation validated against the Admin GraphQL schema):

```
mutation {
  metafieldDefinitionCreate(
    definition: {
      name: "PO reference"
      namespace: "my_app"
      key: "po_reference"
      type: "single_line_text_field"
      ownerType: ORDER
      capabilities: { cartToOrderCopyable: { enabled: true } }
    }
  ) {
    createdDefinition { id namespace key }
    userErrors { field message code }
```

```
  }
}
```

Always read `userErrors`. Shopify's migration guide says the app needs the `write_orders` scope for the metafields to reach the order; the tutorial does not mention it, so confirm it in your app.

STEP 05

Deploy and activate

1. Deploy the version: `shopify app deploy`. With `--no-release` you create the version without releasing it.
2. In the store admin: *Settings > Checkout > Customize checkout*. Pick the step, click *Add app block* and select the extension. Without this step the extension does not show.
3. Place a test order: fill in the field, complete the purchase and open the order's detail page. The value shows in the Metafields section.

> **CAUTION**
> We did not run this extension in a store: we validated the code and the mutation with Shopify's tools, and the configuration follows their example repository. Place the test order before you turn it on for shoppers. With accelerated checkout (Apple Pay, Google Pay) the change can fail, which is why the code checks `canSetCartMetafields`.

WHEN YOU FINISH

Checklist

- [ ] The store is on Shopify Plus if the field goes on information, shipping or payment.
- [ ] `namespace` and `key` match in the `.toml`, in the code and in the order definition.
- [ ] The order definition has copy from cart turned on, and the app has `write_orders`.
- [ ] The code checks `canSetCartMetafields` and saves when the field loses focus.
- [ ] The merchant added the app block in the checkout editor.
- [ ] A test order shows the value under Metafields.
- [ ] Without Plus: ask in the theme's cart with an `input` named `attributes[name]`.